DOI: https://doi.org/10.62985/j.huit_ojs.vol26.no1.397

KHAI THÁC TẬP HỮU ÍCH TRUNG BÌNH CAO VỚI RÀNG BUỘC TƯƠNG QUAN

Nguyễn Thị Thanh Thủy, Nguyễn Văn Lễ*, Mạnh Thiên Lý

Trường Đại học Công Thương Thành phố Hồ Chí Minh

*Email: *lenv@huit.edu.vn*

Ngày nhận bài: 02/5/2024; Ngày nhận bài sửa: 27/9/2024; Ngày chấp nhận đăng: 14/10/2024

TÓM TẮT

Tập hữu ích cao là một trong những khái niệm quan trọng trong bài toán khai thác tập mục. Nhằm khắc phục sự mất cân đối về độ hữu ích giữa các phần tử trong tập mục, khái niệm độ hữu ích trung bình đã được đề xuất. Tuy nhiên, một số tập mục đạt độ hữu ích trung bình cao vẫn tồn tại mức tương quan thấp giữa các phần tử, làm giảm giá trị trong quá trình phân tích kinh doanh. Để khắc phục hạn chế này, bài báo đề xuất phương pháp khai thác tập hữu ích trung bình cao có xét đến yếu tố tương quan thông qua thuật toán CHAU (Correlated High Average Utility). Nghiên cứu tập trung cải tiến công thức tính giá trị chặn trên của độ hữu ích trung bình nhằm tăng khả năng tỉa ứng viên, từ đó nâng cao hiệu suất xử lý của thuật toán. Kết quả thực nghiệm so sánh với phương pháp tiên tiến hiện nay là thuật toán CoHAI trên các bộ dữ liệu có đặc trưng thưa và dày khác nhau như Chainstore, Kosarak, Retail, Accident, Mushroom và Chess cho thấy phương pháp đề xuất đạt hiệu năng tốt hơn xét trên cả thời gian thực thi và mức tiêu thụ bộ nhớ.

Từ khóa: Độ đo tương quan, tập hữu ích trung bình cao tương quan, khai thác CoHAU, ràng buộc tương quan trong tập mục, ngưỡng độ hữu ích tối thiểu.

1. MỞ ĐẦU

Trong lĩnh vực kinh doanh, việc phát hiện các tri thức tiềm ẩn từ dữ liệu bán hàng có vai trò quan trọng trong việc hỗ trợ nhà quản trị xây dựng chiến lược và lựa chọn phương án điều hành phù hợp. Một trong những hướng tiếp cận phổ biến nhằm khai thác các thông tin có giá trị từ dữ liệu giao dịch là bài toán khai thác tập hữu ích cao (High Utility Itemset – HUI). Những nghiên cứu ban đầu về vấn đề này có thể kể đến thuật toán Two-Phase [1] do Liu và các cộng sự giới thiệu vào năm 2005, sau đó nhiều phương pháp tiếp tục được phát triển như UP-Growth [2], HUI-Miner [3], FHM [4], FHN [5], EFIM [6] và HMiner [7]. Mặc dù các phương pháp HUI mang lại lợi ích đáng kể trong phân tích dữ liệu kinh doanh, các tập mặt hàng có sự chênh lệch lớn về giá trị hữu ích giữa các phần tử thường khiến các mục có mức hữu ích thấp trở nên kém giá trị, từ đó làm giảm hiệu quả phân tích. Để khắc phục tình trạng mất cân đối này trong tập kết quả, giá trị độ hữu ích của tập mục được xem xét dưới dạng trung bình. Trên cơ sở đó, nhiều phương pháp khai thác tập hữu ích trung bình cao (High Average Utility Itemset – HAU) đã được đề xuất, tiêu biểu như HAUP-Growth dựa trên cấu trúc cây [8], cùng với các thuật toán MHAI [9] và HAU-Miner [10] sử dụng cấu trúc Utility-List. Cách tiếp cận này góp phần hạn chế hiện tượng mất cân bằng về hữu ích giữa các phần tử trong tập mục. Tuy vậy, trong kết quả khai thác vẫn xuất hiện những tập mục mà mức liên hệ giữa các phần tử chưa đủ chặt chẽ, gây ảnh hưởng đến quá trình phân tích dữ liệu. Năm 2020, Sethi và các cộng sự đã giới thiệu thuật toán CoHAI [11], cho phép khai thác các tập hữu ích trung bình cao đồng thời xét đến mức độ tương quan giữa các phần tử trong tập mục (Correlated High Average Utility itemset – CoHAU). Tuy nhiên, hiệu năng xử lý của phương pháp này vẫn còn hạn chế. Trong nghiên cứu này, thuật toán CHAU được đề xuất nhằm cải thiện hiệu suất khai thác các tập hữu ích trung bình cao có xét đến yếu tố tương quan. Điểm đóng góp chính của nghiên cứu là việc điều chỉnh cách xác định giá trị hữu ích trung bình cực đại (MAU) kết hợp với nhiều chiến lược cắt tỉa hiệu quả, qua đó nâng cao hiệu năng thực thi của thuật toán. Kết quả thực nghiệm so sánh với CoHAI cho thấy CHAU đạt kết quả tốt hơn về cả thời gian xử lý lẫn dung lượng bộ nhớ tiêu thụ.

2. CÁC CÔNG TRÌNH LIÊN QUAN

Các nghiên cứu về khai thác tập hữu ích cao đã mang lại nhiều ứng dụng thực tiễn, đặc biệt trong phân tích dữ liệu kinh doanh. Những phương pháp này cho phép rút trích tri thức từ các tập dữ liệu quy mô lớn với hiệu suất xử lý ngày càng được cải thiện. Thuật toán Two-Phase [1] được giới thiệu vào năm 2005 thực hiện khai thác tập hữu ích cao dựa trên hai bước xử lý. Bước đầu quét cơ sở dữ liệu nhằm xác định tập ứng viên, sau đó quét lần hai để xác định những tập mục thực sự đạt ngưỡng hữu ích cao. Việc xử lý dữ liệu qua hai bước khiến phương pháp này kém hiệu quả khi áp dụng trên các tập dữ liệu lớn. Đến năm 2012, Liu và cộng sự giới thiệu cấu trúc Utility-List và thuật toán HUI-Miner [3] cho phép khai thác tập hữu ích cao chỉ trong một pha giúp cải thiện đáng kể tốc độ xử lý. Utility-List sau đó trở thành nền tảng cho nhiều thuật toán một pha được đề xuất tiếp theo. Năm 2014, Fournier-Viger đề xuất chiến lược tỉa mới với cấu trúc EUCS cùng với thuật toán FHM [4] đã cho kết quả thực thi vượt trội hơn HUI-Miner trên các cơ sở dữ liệu thưa. Đến năm 2016, nhóm tác giả này tiếp tục mở rộng FHM bằng phiên bản FHM+ [12], bổ sung thêm ràng buộc về kích thước tập mục trong quá trình khai thác. Năm 2017, Zida và cộng sự áp dụng cơ chế chiếu dữ liệu giao dịch kết hợp phương pháp gộp trong thuật toán EFIM [6], chứng minh được hiệu quả cao trong việc tìm kiếm tập HUI. Tiếp đó, thuật toán HMiner [7] được đề xuất bởi Krishnamoorthy cho kết quả vượt trội hơn so với các thuật toán trước đây. Thuật toán sử dụng cấu trúc dữ liệu mới CUL kết hợp nhiều chiến lược cắt tỉa, giúp giảm đáng kể không gian tìm kiếm trong quá trình khai thác.

Song song với hướng nghiên cứu trên, nhiều công trình đã mở rộng bài toán HUI theo các hướng tiếp cận khác nhau, trong đó việc khai thác tập hữu ích trung bình cao đã được nghiên cứu nhằm hạn chế tình trạng mất cân đối về độ hữu ích giữa các mục trong tập kết quả, qua đó hỗ trợ việc phân tích và đánh giá dữ liệu chính xác hơn. Năm 2009, Hong và cộng sự đề xuất phương pháp khai thác tập mục có hữu ích trung bình cao [13], tuy nhiên phương pháp này vẫn sử dụng cơ chế hai pha tương tự Two-Phase nên hiệu quả thực thi còn hạn chế. Đến năm 2010, thuật toán HAUP-Growth được giới thiệu bởi nhóm tác giả Lin và cộng sự, hoạt động dựa trên cấu trúc cây HAUP-tree [8] để tổ chức lưu trữ thông tin trong quá trình khai thác HAU. Năm 2016, nhóm tác giả tiếp tục đề xuất HAUI-Miner [14] dựa trên Utility-List để tìm tập hữu ích trung bình cao trong một lần quét dữ liệu, do thực thi trên một pha nên hiệu suất thực thi tốt hơn so với HAUP-Growth. Năm 2017, thuật toán MHAI [9] được giới thiệu với chiến lược cắt tỉa sử dụng giá trị hữu ích trung bình cực đại (MAU) trong giao dịch, giúp thu hẹp đáng kể không gian tìm kiếm. Sau đó, năm 2021, Pham và cộng sự [10] cải tiến công thức tính MAU và chứng minh hiệu quả của phương pháp trong thực nghiệm.

Nhằm tăng mức độ liên hệ giữa các mục trong kết quả khai thác, tính tương quan được Gan và cộng sự [15] giới thiệu kết hợp với bài toán khai thác tập hữu ích cao để khai thác tập hữu ích cao tương quan. Tiếp đó, nhóm tác giả cải tiến phương pháp bằng việc sử dụng Utility-List và phát triển thuật toán CoUPM [16]. Năm 2020, thuật toán CoHUI-Miner [17] được giới thiệu nhằm kết hợp phương pháp chiếu dữ liệu cùng nhiều kỹ thuật cắt tỉa đã cải thiện đáng kể hiệu suất xử lý trong bài toán khai thác HUI có xét tương quan. Gần đây, Sethi và cộng sự đã kết hợp yếu tố tương quan với phương pháp tính hữu ích trung bình cao, từ đó đề xuất thuật toán CoHAI [11] dựa trên Utility-List và độ đo Kulc để tìm các tập mục hữu ích trung bình cao có tương quan (Correlated High Average Utility Itemset – CoHAU).

3. ĐỊNH NGHĨA VÀ KÝ HIỆU

Một cơ sở dữ liệu giao dịch được ký hiệu D gồm tập hợp các giao dịch T_j với $j \in \{1, \dots, N\}$ và một tập I chứa m mục phân biệt trong D. Mỗi giao dịch T_j chứa t mục $\{x_1, x_2, \dots, x_t\}$ với $x_t \in I$ được trình bày trong Bảng 1. Một mục $x_t \in T_j$ có độ hữu ích là $U(x_t, T_j) = p(x_t) * q(x_t, T_j)$ với $p(x_t)$ là lợi nhuận (profit) của mặt hàng x_t và $q(x_t, T_j)$ là số lượng của x_t trong giao dịch T_j. Một tập mục $X \subseteq T_j$ có độ hữu ích $U(X, T_j) = \sum_{x_t \in X \wedge X \subseteq T_j} U(x_t, T_j)$. Trong toàn bộ cơ sở dữ liệu D, $U(X) = \sum_{X \subseteq T_j \wedge T_j \in D} U(X, T_j)$. Ví dụ dựa vào cơ sở dữ liệu D (Bảng 1), mục a trong T_1 có độ hữu ích là $U(a, T_1) = p(a) * q(a, T_1) = 2 * 6 = 12$. Tập mục $\{a, b\}$ có độ hữu ích $U(\{a, b\}) = U(\{a, b\}, T_1) + U(\{a, b\}, T_4) + U(\{a, b\}, T_6) = 28 + 14 + 14 = 56$.

Định nghĩa 1. Độ hữu ích cực đại (Maximum Utility - MaxU) trong một giao dịch T_j là độ hữu ích của một mục $x_t \in T_j$ có giá trị lớn nhất so với độ hữu ích của các mục khác trong T_j, ký hiệu

$MaxU(T_j) = \max\{U(x_t, T_j)\}$ [9]. Ví dụ xét $T_3 \in D$ (Bảng 1), $MaxU(T_3) = \max\{U(c, T_3), U(d, T_3), U(e, T_3)\} = \max\{30, 6, 3\} = 30$. Giá trị $MaxU$ của các giao dịch được trình bày ở cột 3 trên Bảng 1.

Định nghĩa 2. Độ hữu ích trung bình của tập mục $X = \{x_1, x_2, \dots, x_k\}$ trên một giao dịch T_j ký hiệu $AU(X, T_j) = \frac{U(X,T_j)}{k}$. Độ hữu ích trung bình của X trên D ký hiệu $AU(X) = \sum_{X \subseteq T_j \wedge T_j \in D} AU(X, T_j)$ [9]. Ví dụ xét tập mục $\{a, b\} \subseteq T_1$, $AU(\{a, b\}, T_1) = \frac{U(\{a,b\},T_1)}{2} = \frac{28}{2} = 14$. Độ hữu ích trung bình của $\{a, b\}$ trên D có giá trị $AU(\{a, b\}) = AU(\{a, b\}, T_1) + AU(\{a, b\}, T_4) + AU(\{a, b\}, T_6) = 14 + 7 + 7 = 28$

Bảng 1. Cơ sở dữ liệu giao dịch D

TID	Giao dịch	Maximum Utility (MaxU)
1	(a,6) (b,4) (c,1) (d,6)	16
2	(b,5) (d,4) (f,2)	20
3	(c,6) (d,3) (e,1)	30
4	(a,5) (b,1) (d,6)	12
5	(a,2) (c,1) (d,2) (e,3) (f,4)	9
6	(a,3) (b,2) (c,2) (d,5) (e,3)	10
7	(a,5) (c,4) (d,6) (e,2)	20

Bảng 2. Giá trị lợi nhuận

Mục	a	b	c	d	e	f
Lợi nhuận	2	4	5	2	3	1

Định nghĩa 3. Tập mục Y được gọi là tập hữu ích trung bình cao (HAU) nếu $AU(Y) \geq minUtil$ [9], với minUtil là ngưỡng tối tiểu của độ hữu ích. Ví dụ Với $minUtil = 30$ thì tập mục $\{a, b\}$ không là HAU vì $AU(ab) = 28 < minUtil = 30$.

Định nghĩa 4. Chặn trên của độ hữu ích trung bình của tập mục X trong D được định nghĩa là $UB(X) = \sum_{X \subseteq T_j} MaxU(T_j)$ [9]. Giá trị UB của các mục $x_k \in I$ được trình bày trong Bảng 3.

Chiến lược tỉa 1. Xét Y là tập được mở rộng từ tập X. Nếu $UB(X) < MinUtil$ thì mọi tập Y đều không phải tập HAU. Khi đó X sẽ bị cắt tỉa khỏi không gian tìm kiếm. Trường hợp X là tập mục một phần tử $x_k \in I$ thì x_k sẽ bị loại khỏi D [9]. Ví dụ $UB(f) = MU(T_2) + MU(T_5) = 20 + 9 = 29 < minUtil = 30$ nên f sẽ bị loại bỏ khỏi D.

Định nghĩa 5. Số giao dịch chứa tập mục X trong D được gọi là độ hỗ trợ của X, ký hiệu $SUP(X)$ [17]. Ví dụ xét tập mục $\{c, d\}$, ta có $SUP(cd) = 5$ vì $\{c, d\}$ xuất hiện trong cả 5 giao dịch là T_1, T_3, T_5, T_6, T_7. Bảng 3 trình bày độ hỗ trợ của các mục một phần tử.

Bảng 3. Độ hỗ trợ (SUP) và chặn trên (UB) các mục đơn

Mục	a	b	c	d	e	f
SUP	5	4	5	7	4	2
UB	67	58	85	117	69	29

Bảng 4. Cơ sở dữ liệu D được sắp thứ tự và loại bỏ mục f

TID	Giao dịch	MU
1	(b,4) (a,6) (c,1) (d,6)	16
2	(b,5) (d,4)	20
3	(e,1) (c,6) (d,3)	30
4	(b,1) (a,5) (d,6)	12
5	(e,3) (a,2) (c,1) (d,2)	9
6	(b,2) (e,3) (a,3) (c,2) (d,5)	10
7	(e,2) (a,5) (c,4) (d,6)	20

Định nghĩa 6. Xét $x_i, x_j \in I$, nếu $SUP(x_i) < SUP(x_j)$ thì x_i được sắp thứ tự đứng trước x_j trong thứ tự toàn phần của các mục trong D, ký hiệu $x_i \prec x_j$. Nếu $SUP(x_i) = SUP(x_j)$ thì tính theo thứ tự Alphabet [17]. Với giá trị SUP của các mục trong Bảng 3 thì thứ tự các mục trong D là $f \prec b \prec e \prec a \prec c \prec d$. Sau khi sắp thứ tự các mục và loại bỏ mục f trong D (vì vì $UB(f) < minUtil$) thì được cơ sở dữ liệu mới như trong Bảng 4.

Định nghĩa 7. Xét tập mục $X = \{x_1, x_2, \dots, x_k\}$, mức độ tương quan giữa các phần tử trong X ký hiệu $kulc(X) = \frac{\text{SUP(X)}}{k}\left(\frac{1}{SUP(x_1)} + \frac{1}{SUP(x_2)} + \cdots + \frac{1}{SUP(x_k)}\right)$. Trong đó, $kulc(X) \in [0,1]$ [15] Những tập mục một phần tử sẽ có giá trị $kulc = 1$. Ví dụ $kulc(bc) = \frac{\text{SUP(bc)}}{2}\left(\frac{1}{SUP(\text{b})} + \frac{1}{SUP(\text{c})}\right) = \frac{2}{2}\left(\frac{1}{4} + \frac{1}{5}\right) = 0{,}45$

Tính chất. Xét tập mục Y mở rộng từ X, ta có $Kulc(X) >= Kulc(Y)$ [15]. Ví dụ xét tập mục $\{b, c, d\}$ là tập mở rộng của $\{b, c\}$, $Kulc(bcd) = \frac{SUP(bcd)}{3}\left(\frac{1}{SUP(\text{b})} + \frac{1}{SUP(\text{c})} + \frac{1}{SUP(\text{d})}\right) = \frac{2}{3}\left(\frac{1}{4} + \frac{1}{5} + \frac{1}{7}\right) = 0{,}39 < Kulc(bc) = 0{,}45$.

Chiến lược tỉa 2. Nếu $Kulc(X) < minCor$ thì mọi tập Y mở rộng từ X đều có $Kulc(Y) \leq Kulc(X) < minCor$ (Tính chất trên) nên dừng mở rộng với tập X [16]. Ví dụ với $minCor = 0{,}5$ thì tập mục $\{b, c\}$ có $Kulc(bc) = 0{,}45 < minCor = 0{,}5$ nên dừng mở rộng từ tập $\{b, c\}$.

Định nghĩa 8. Xét X là một tập mục, nếu $kulc(X) >= minCor$ *và* $AU(X) >= minUtil$ thì X là tập hữu ích trung bình cao có tính tương quan [11]. Trong đó, $minCor \in [0,1]$ là ngưỡng tương quan tối thiểu và $minUtil$ là ngưỡng độ hữu ích tối thiểu. Ví dụ xét tập mục $\{c, d\}$ trong D (Bảng 4) có $AU(cd) = 57$, $Kulc(cd) = 0{,}86$. Với giá trị ngưỡng $minCor = 0{,}5$ và $minUtil = 30$ thì $\{c, d\}$ là tập hữu ích trung bình cao có tính tương quan.

Định nghĩa 9. Xét tập mục $X \subset T_j$, một mục y được gọi là phía sau X ký hiệu $X \prec y$ nếu $\forall x_i \in X, x_i \prec y$ theo thứ tự toàn phần trình bày trong Định nghĩa 6. Độ hữu ích lớn nhất phía sau $X \subset T_j$ ký hiệu $MRU(X, T_j) = Max\{U(y) | X \prec y \text{ với } y \in T_j\}$ [9].

Ví dụ xét tập mục $\{b, a\} \subset T_6 \in D$ (Bảng 4), có $\{b, a\} \prec c$ và $\{b, a\} \prec d$. $MRU(ba, T_6) = Max\{U(c, T_6), U(d, T_6)\} = Max\{10, 10\} = 10$.

4. THUẬT TOÁN ĐỀ XUẤT

4.1. Độ hữu ích trung bình lớn nhất

Trong bài toán khai thác tập hữu ích trung bình cao, giá trị độ hữu ích trung bình lớn nhất có ảnh hưởng lớn đến kết quả khai thác. Giá trị này càng nhỏ thì khả năng cắt giảm không gian tìm kiếm càng tốt dẫn đến hiệu suất thực thi của thuật toán cao hơn. Do đó, các công trình nghiên cứu về khai thác HAU đã đưa ra nhiều đề xuất với công thức tính giá trị này một cách tối ưu nhất có thể.

Định nghĩa 10. Một mục y được gọi là vượt trội so với tập mục $X \subset T_j$ nếu $X \prec y$ và $AU(X, T_j) \leq U(y, T_j)$. Số mục vượt trội tập X ký hiệu $S(X, T_j)$. Ví dụ: Xét giao dịch $T_6 \in D$ ở Bảng 4, Với tập mục $\{b, e\}$, ta có $AU(be, T_6) = 8{,}5$. Những mục phía sau $\{b, e\}$ gồm a, c và d (vì $\{b, e\} \prec a, \{b, e\} \prec c$, $\{b, e\} \prec d$), $U(a, T_6) = 6 < AU(be, T_6) = 8{,}5$ nên a không phải là một mục vượt trội tập $\{b, e\}$ trong T_6, $U(c, T_6) = 10$ và $U(d, T_6) = 10$ đều lớn hơn $AU(be, T_6) = 8{,}5$ nên c và d là các mục vượt trội tập $\{b, e\}$. Số phần tử vượt trội tập $\{b, e\}$ trong T_6 là $S(be, T_6) = 2$.

Định nghĩa 11. Độ hữu ích trung bình lớn nhất của tập mục $X \subseteq T_j$ ký hiệu $MAU(X, T_j)$.

$$MAU(X, T_j) = \begin{cases} \dfrac{U(X, T_j) + MRU(X, T_j) * S(X, T_j)}{|X| + S(X, T_j)}, \text{nếu } AU(X, T_j) < MRU(X, T_j) \\ \dfrac{U(X, T_j) + MRU(X, T_j)}{|X| + 1}, \text{nếu } AU(X, T_j) \geq MRU(X, T_j) > 0 \\ 0, MRU(X, T_j) = 0 \end{cases}$$

Độ hữu ích trung bình lớn nhất của tập mục X trong toàn bộ cơ sở dữ liệu D được định nghĩa $MAU(X) = \sum_{X \subseteq T_j \in D} MAU(X, T_j)$.

Xét tập mục Y phía sau tập mục X trong T_j nghĩa là $X \prec Y$. Khi mở rộng tập mục X thành XY, thì $AU(XY, T_j)$ có thể tăng so với $AU(X, T_j)$ nếu như $AU(X, T_j) < MRU(X, T_j)$ và có những phần tử $y \in Y' \subseteq Y$ với $U(y, T_j) > AU(X, T_j)$. Giá trị $AU(XY, T_j)$ sẽ tăng cực đại nếu $U(y) = MRU(X, T_j)$. Mặc khác, $|Y'| \leq S(X)$ nên $AU(XY, T_j) \leq MAU(X, T_j)$. Trường hợp $AU(X, T_j) \geq MRU(X, T_j)$ thì mọi phần tử $y \in Y$ đều có $U(y, T_j) \leq AU(X, T_j)$ nên $AU(XY, T_j)$ có xu hướng giảm so với $AU(X, T_j)$. Giá trị $AU(XY, T_j)$ đạt cực đại khi $\forall y \in Y, U(y, T_j) = MRU(X, T_j)$ nhưng $MRU(X, T_j) < AU(X, T_j)$ nên $AU(XY, T_j) \leq MAU(X, T_j)$.

Ví dụ: Xét tập $\{b,e\} \subset T_6$ trong cơ sở dữ liệu D ở Bảng 4, $MRU(be,T_6) = 10 > AU(be,T_6) = 8{,}5$, $S(be,T_6) = 2$ nên $MAU(be,T_6) = \frac{U(be,T_6)+MRU(be,T_6)*S(be,T_6)}{|be|+S(be,T_6)} = \frac{17+10*2}{2+2} = 9{,}25$ là độ hữu ích trung bình lớn nhất khi mở rộng từ $\{b,e\} \subset T_6$. Những tập có thể mở rộng từ $\{b,e\}$ trong T_6 là $\{b,e,a\},\{b,e,c\},\{b,e,d\},\{b,e,a,c\},\{b,e,a,d\},\{b,e,a,c,d\}$. Độ hữu ích trung bình của các tập này có giá trị như sau: $AU(bea,T_6) = 7{,}67, AU(bec,T_6) = 9,\ AU(bed,T_6) = 9, AU(beac,T_6) = 8{,}25, AU(bead,T_6) = 8{,}25, AU(beacd,T_6) = 8{,}6.$

Có thể thấy rằng $MAU(be)$ luôn lớn hơn AU của các tập mở rộng từ $\{b,e\}$. Do đó nếu $MAU(be) < minUtil$ thì mọi tập mở rộng từ $\{b,e\}$ sẽ bị loại bỏ khỏi không gian tìm kiếm mà không cần phải mở rộng.

Chiến lược tỉa 3. Xét tập mục X trong D, nếu $MAU(X) < minUtil$ thì $\forall Y$ mở rộng từ X đều có $AU(Y) \leq MAU(X) < minUtil$ nên Y không phải là tập $CoHAU$ [11]. Ví dụ: $MAU(be) = 9{,}25 < minUtil = 30$ nên mọi tập mở rộng từ $\{b,e\}$ như $\{b,e,a\},\{b,e,c\},\{b,e,d\},\{b,e,a,c\},\{b,e,a,d\},\{b,e,a,c,d\}$ đều không phải $CoHAU$.

Nghiên cứu này cải tiến phương pháp tính giá trị MAU trong trường hợp $AU(X,T_j) < MRU(X,T_j)$ bằng việc sử dụng giá trị $S(X,T_j)$ thay cho giá trị $mn(X)$ trong thuật toán MHAI [9] và giá trị $ri(X,T_j)$ trong thuật toán HAU-Miner [10]. Trong đó $mn(X)$ là chiều dài lớn nhất của phần giao dịch phía sau X xét trên các $T_j \supset X$, $ri(X,T_j)$ là chiều dài của phần giao dịch T_j phía sau X, $S(X,T_j)$ là chiều dài của phần giao dịch T_j sau X nhưng chỉ xét những phần $x_i \in T_j$ có $U(x_i,T_j) \geq AU(X,T_j)$. Do đó ta có $S(X,T_j) \leq ri(X,T_j) \leq mn(X)$ và tương tự cho độ hữu ích trung bình lớn nhất $MAU_{CHAU} \leq MAU_{HAU-Miner} \leq MAU_{MHAI}$.

Ví dụ: Xét tập $\{b,e\} \subset T_6$ thì $S(be,T_6) = 2$, $ri(be,T_6) = 3$, $mn(be)=3$, $MAU_{CHAU}(be,T_6) = 9{,}25$, $MAU_{HAU-Miner}(be,T_6) = MAU_{MHAI}(be,T_6) = 9{,}4$.

Trong thuật toán CoHAI [11], Sethi và cộng sự đề xuất cách tính chặn trên độ hữu ích trung bình của tập mục $X \subseteq T_j$ thay cho $MAU(X,T_j)$ bằng công thức:

$lub(X,T_j) = \frac{U(X,T_j)+|X|*MRU(X,T_j)}{|X|} = \frac{U(X,T_j)}{|X|} + MRU(X,T_j)$. Ta có giá trị $lub(X,T_j)$ luôn lớn hơn giá trị $MAU(X,T_j)$ trong mọi trường hợp giá trị của $MRU(X,T_j)$. Do đó chiến lược tỉa dựa trên MAU sẽ hiệu quả hơn chiến lược tỉa dựa trên lub.

Chứng minh: Ta có $lub(X,T_j) = \frac{U(X,T_j)}{|X|} + MRU(X,T_j)$ [11]

Xét các trường hợp của công thức tính $MAU(X,T_j)$ trong Định nghĩa 11:

Trường hợp 1: Với $MRU(X,T_j) = 0$, ta có $MAU(X,T_j) = 0 < lub(X,T_j) = \frac{U(X,T_j)}{|X|}$ (1)

Trường hợp 2: $AU(X,T_j) \geq MRU(X,T_j) > 0$, có $MAU(X,T_j) = \frac{U(X,T_j)+MRU(X,T_j)}{|X|+1} = \frac{U(X,T_j)}{|X|+1} + \frac{MRU(X,T_j)}{|X|+1}$

Vì $\frac{U(X,T_j)}{|X|+1} < \frac{U(X,T_j)}{|X|}$ và $\frac{MRU(X,T_j)}{|X|+1} < MRU(X,T_j)$ nên $\frac{U(X,T_j)}{|X|+1} + \frac{MRU(X,T_j)}{|X|+1} < \frac{U(X,T_j)}{|X|} + MRU(X,T_j)$ hay $MAU(X,T_j) < Lub(X,T_j)$ (2)

Trường hợp 3: $AU(X,T_j) < MRU(X,T_j)$, ta có:

$MAU(X,T_j) = \frac{U(X,T_j)+MRU(X,T_j)*S(X,T_j)}{|X|+S(X,T_j)} = \frac{U(X,T_j)}{|X|+S(X,T_j)} + MRU(X,T_j) * \frac{S(X,T_j)}{|X|+S(X,T_j)}$

Vì $\frac{S(X,T_j)}{|X|+S(X,T_j)} < 1$ do đó $MRU(X,T_j) * \frac{S(X,T_j)}{|X|+S(X,T_j)} < MRU(X,T_j)$.

Mặc khác $\frac{U(X,T_j)}{|X|+S(X,T_j)} < \frac{U(X,T_j)}{|X|}$ nên $\frac{U(X,T_j)}{|X|+S(X,T_j)} + MRU(X,T_j) * \frac{S(X,T_j)}{|X|+S(X,T_j)} < \frac{U(X,T_j)}{|X|} + MRU(X,T_j)$ hay $MAU(X,T_j) < Lub(X,T_j)$ (3)

Từ (1), (2) & (3) suy ra $MAU(X,T_j) < lub(X,T_j)$ với mọi giá trị của $MRU(X,T_j)$.

Ví dụ: Xét tập $\{b,e\} \subset T_6$, ta có $lub(be,T_6) = \frac{U(be,T_6)}{2} + MRU(be,T_6) = 8{,}5 + 10 = 18{,}5 > MAU(be,T_6) = 9{,}25$. Có thể thấy rằng công thức tính MAU chúng tôi đề xuất có giá trị MAU nhỏ hơn đáng kể so với giá trị lub nên khả năng tỉa tốt hơn, dẫn đến nâng cao hiệu suất khai thác tập mục.

4.2. Cấu trúc dữ liệu C-List

Để phù hợp với bài toán khai thác tập CoHAU, chúng tôi đề xuất cấu trúc C-List cải tiến từ cấu trúc Utility-List [3]. Trong đó, mỗi bộ trong C-List của tập mục X gồm có bốn thành phần $< Tid, U(X), MRU(X), Pos >$ với Tid là giao dịch $T_j \supset X, U(X)$ là giá trị độ hữu ích của $X, MRU(X)$ là độ hữu ích lớn nhất sau X. Giá trị Pos được dùng để tính $S(X, T_j)$ trong công thức tính $MAU\ (X)$ (Định nghĩa 11).

Tập {b} MAU=46,8

Tid	U	MRU	Pos
1	16	12	1
2	20	8	1
4	4	12	1
6	8	10	1

Tập {e} MAU=54,2

Tid	U	MRU	Pos
3	3	30	1
5	9	5	1
6	9	10	2
7	6	20	1

Tập {a} MAU=53

Tid	U	MRU	Pos
1	12	12	2
4	10	12	2
5	4	5	2
6	6	10	3
7	10	20	2

Tập {c} MAU=57

Tid	U	MRU	Pos
1	5	12	3
3	30	6	2
5	5	4	3
6	10	10	4
7	20	12	3

Tập {d} MAU=0

Tid	U	MRU	Pos
1	12	0	4
2	8	0	2
3	6	0	3
4	12	0	3
5	4	0	4
6	10	0	5
7	12	0	4

Tập {b,e} MAU=9,25

Tid	U	MRU	Pos
6	17	10	2

Tập {b,a} MAU=30,5

Tid	U	MRU	Pos
1	28	12	2
4	14	12	2
6	14	10	3

Tập {b,c} MAU=20,3

Tid	U	MRU	Pos
1	21	12	3
6	18	10	4

Tập {b,d} MAU=0

Tid	U	MRU	Pos
1	28	0	4
2	28	0	2
4	16	0	3
6	18	0	5

Tập {b,e,a} MAU=8,6

Tid	U	MRU	Pos
6	23	10	3

Tập {b,e,c} MAU=9,25

Tid	U	MRU	Pos
6	27	10	4

Tập {b,e,d} MAU=0

Tid	U	MRU	Pos
6	27	0	5

Hình 1. Danh sách C-List mở rộng ba cấp

Hình 1 trình bày danh sách C-List mở rộng ba cấp được tạo từ cơ sở dữ liệu D ở Bảng 4. Các C-List một phần tử (Cấp 1) được sắp xếp từ trái sang phải dựa trên thứ tự toàn phần của các mục trong cơ sở dữ liệu D. Trong danh sách này, mục f bị loại bỏ vì có $UB(f) = 29 < minUtil = 30$. Việc mở rộng tập mục được thực hiện bằng phương pháp duyệt theo chiều sâu. Danh sách C-List cấp một sẽ được khởi tạo lần đầu khi quét cơ sở dữ liệu D như sau: Xét giao dịch $T_1 = \{(b,4),(a,6),(c,1),(d,6)\}$, duyệt lần lượt từng mục từ trái qua phải. Xét mục mục $(b,4)$ ta có $U(b,T_1) = 4 * 4 = 16, MRU(b,T_1) = 12$, phần tử bắt đầu phía sau b là phần tử a có vị trí trong giao dịch T_1 là Pos=1, như vậy bộ <T_1, 16, 12, 1> được thêm vào C-List b. Tương tự cho các mục khác trong giao dịch T_1. Lần lượt duyệt qua tất cả các giao dịch T_j còn lại trong D sẽ được danh sách C-List như trong Hình 1.

Danh sách C-List cấp k (k>1) được thực hiện bằng cách kết hợp các C-List cấp k-1 theo thứ tự từ trái qua phải. Đối với các C-List cấp k=2 thì các giá trị trong C-List kết hợp được tính như sau: xét Z là C-List kết hợp từ hai C-List X, Y, ta có $Z.Tid.U = X.Tid.U + Y.Tid.U, Z.Tid.MRU = Y.Tid.MRU$ và $Z.Tid.Pos = Y.Tid.Pos$. Ví dụ xét Z = C-List $\{b,e\}$ được kết hợp từ X = C-List $\{b\}$ và Y= C-List $\{e\}$. Trong hai C-List $\{b\}$ và $\{e\}$ chỉ có thể kết hợp được bộ T_6 vì bộ này xuất hiện trong cả 2 C-List. Ta có $Z.\ T_6.U = X.T_6.U + Y.T_6.U$=8+9=17, $Z.T_6.MRU = Y.T_6.MRU = 10,\ Z.T_6.Pos = Y.T_6.Pos = 2$. Ta có bộ <$T_6$, 17, 10, 2> được thêm vào C-List $\{b,e\}$ như hình 1. Thực hiện tương tự cho các C-List khác ở cấp 2.

Đối với các C-List cấp k>2, cách tính giá trị độ hữu ích thực hiện như sau: Xét Q là một C-List tiền tố ở cấp k-1, X và Y là hai C-List đang kết hợp ở cấp k, Z là C-List kết hợp X và Y. Độ hữu ích của các bộ trong Z được tính là $Z.Tid.U = (X.Tid.U + Y.Tid.U) - Q.Tid.U$. Các giá trị còn lại như MRU, Pos được tính tương tự như cấp k=2. Ví dụ: Xét X=C-List $\{b,e\}$ và Y=C-List $\{b,a\}$ có tiền tố Q=C-List $\{b\}$, Z=C-List $\{b,e,a\}$ là kết quả sau khi kết hợp C-List X và C-List Y. Ta có $Z.T_6.U = (X.T_6.U + Y.T_6.U) - Q.T_6.U = 17 + 14 - 8 = 23, Z.T_6.MRU = Y.T_6.MRU = 10, Z.T_6.Pos = Y.T_6.Pos = 3$. Như vậy, một bộ $< T_6, 23, 10, 3 >$ được thêm vào C-List kết hợp. Các C-List tiếp theo được thực hiện tương tự để được danh sách C-List cấp k=3.

4.3. Thuật toán đề xuất

Chúng tôi đề xuất thuật toán $CHAU$ (Correlated-High Average Utility) để khai thác tập CoHAU. Đầu vào của thuật toán gồm một cơ sở dữ liệu giao dịch D, ngưỡng minUtil và minCor. Đầu ra là tập CoHAUs chứa các tập hữu ích trung bình cao có tính tương quan. Bước đầu, tính các giá trị UB và độ hỗ trợ SUP cho từng mục đơn x trong D (Dòng 1) đồng thời kiểm tra điều kiện nếu $UB(x) \geq minUtil$ thì thêm x vào tập mục $\mathcal{I}$, ngược lại loại mục đơn x khỏi D. Bước tiếp theo, sắp xếp tập $\mathcal{I}$ tăng theo độ hỗ trợ Support, đồng thời sắp D theo thứ tự $\mathcal{I}$ (Dòng 2 và 3). Tại dòng 4, mỗi mục $x \in \mathcal{I}$ được khởi tạo

một C-List tương ứng. Dòng 5 xây dựng cấu trúc EUCS để tỉa không gian tìm kiếm trong thuật toán MiningCoHAU trình bày ở dòng 6.

Thuật toán CHAU

Đầu vào: Cơ sở dữ liệu D, ngưỡng $minUtil$, ngưỡng $minCor$.
Đầu ra: Tập $CoHAUs$ (Các tập hữu ích trung bình cao có tính tương quan).

1. Duyệt D lần đầu, tính $UB(x)$ và $SUP(x)$ với $\forall x \in I$.
2. Tính $\mathcal{I} = \{x \in I \mid UB(x) \geq minUtil\}$, nếu $x \notin \mathcal{I}$, loại x khỏi D.// Chiến lược tỉa 1
3. Sắp xếp tập $\mathcal{I}$ theo thứ tự tăng của SUP, sắp các mục $x \in D$ theo thứ tự của $\mathcal{I}$.
4. Xây dựng *C-List* cho mỗi mục đơn $x \in \mathcal{I}$
5. Xây dựng cấu trúc $EUCS$
6. MiningCoHAU(∅, C-List, minUtil, minCor, EUCS)

Chiến lược tỉa 4. Xét các cặp mục đơn $x_i, x_j \in I$, nếu $UB(x_i, x_j) < minUtil$ thì ngừng mở rộng tập mục $\{x_i, x_j\}$ vì khi đó tập mục $\{x_i, x_j\}$ và các tập mục mở rộng của $\{x_i, x_j\}$ đều không là tập $CoHAU$. Xét cơ sở dữ liệu D ở Bảng 4, cấu trúc EUCS được khởi tạo gồm các giá trị là một ma trận tam giác trên với $EUCS(x, y) = UB(x, y)$. Vi dụ: $EUCS(b, e) = 10, < minUtil = 50$ nên tập $\{b, e\}$ không phải là $CoHAU$ và dừng mở rộng với tập $\{b, e\}$.

	e	a	c	d
b	10	38	26	58
e	0	39	69	69
a	0	0	55	67
c	0	0	0	85

Hình 2. Cấu trúc EUCS

Thuật toán *MiningCoHAU* có đầu vào gồm danh sách C-List CLs và một C-List tiền tố Q cùng với ngưỡng $minUtil$ và ngưỡng $minCor$. Đầu ra của thuật toán là tập CoHAUs (các tập hữu ích trung bình cao có tính tương quan). Từ dòng 1 đến dòng 4 thực hiện vòng lặp for để duyệt qua lần lượt các C-List Y trong danh sách đầu vào CLs. Thực hiện tính độ hữu ích trung bình và giá trị $Kulc$ của tập mục ứng với C-List Y, kiểm tra nếu $AU(Y) \geq minUtil \wedge Kulc(Y) \geq minCor$ thì đưa Y vào tập kết quả $CoHAUs$. Dòng 5 sử dụng chiến lược tỉa 2 và 3 để kiểm tra tập Y có khả năng mở rộng hay không bằng cách kiểm tra nếu thoả điều kiện kép là $Kulc(Y) \geq minCor$ và $MAU(Y) \geq minUtil$ thì tiếp tục mở rộng. Dòng 6 khởi tạo tập $exCLs$ để chứa danh sách các C-List kết quả mở rộng từ danh sách CLs đầu vào. Dòng 7 lần lượt xét các C-List Z phía sau Y trong CLs, sử dụng chiến lược tỉa 4 bằng cách kiểm tra nếu $EUCS(Y, Z) \geq minUtil$ thì gọi thuật toán *CoHAUConstruct* để kết hợp hai C-List Y, Z và đưa C-List kết quả vào tập $exRLs$ được trình bày ở dòng 8,9. Dòng 12 tiếp tục mở rộng không gian tìm kiếm bằng cách gọi đệ quy thuật toán *MiningCoHAU*.

Thuật toán đệ quy MiningCoHAU

Đầu vào: Q: C-List làm tiền tố, CLs: danh sách các C-List có Q là tiền tố, ngưỡng $minUtil$, ngưỡng $minCor$.

Đầu ra: Tập $CoHAUs$ (các tập hữu ích trung bình cao có tính tương quan).

1. **foreach** C-List Y in CLs **do**
2. **if** $AU(Y) \geq minUtil \wedge Kulc(Y) \geq minCor$ **then**
3. $CoHAUs \leftarrow Y$
4. **end if**
5. **if** $(Kulc(Y) \geq minCor \wedge MAU(Y) \geq minUtil)$ //Chiến lược tỉa 2 và 3
6. $exCLs = \emptyset$
7. **foreach** Z *in* $CLs \wedge Y \prec Z$ **do**
8. **if** $EUCS(Y, Z) \geq minUtil$ **then** //Chiến lược tỉa 4
9. $exRLs \leftarrow CoHAU\text{Construct}(Q, Y, Z)$,
10. **end if**
11. **end for**
12. $MiningCoHAU(Y, exCLs, minUtil, minCor)$,
13. **end if**
14. **end for**

Thuật toán *CoHAUConstruct* dùng để xây dựng một C-List mới từ hai C-List đã có, dữ liệu đầu vào là hai C-List X và Y, một C-List tiền tố Q và ngưỡng $minUtil$. Kết quả đầu ra là C-List Z kết hợp giữa X và Y. Dòng 1 thực hiện gán $MAU_{LA} = MAU(X)$ để khởi tạo giá trị đầu tiên trong chiến lược tỉa LA. Vòng lặp ở dòng 2 xét các bộ t_x thuộc C-List X và kiểm tra điều kiện nếu xuất hiện bộ t_y trong C-List Y mà $t_x.Tid = t_y.Tid$ thì tạo bộ kết quả $t_{x,y} \leftarrow < t_x.Tid,\ t_x.U + t_y.U,\ t_y.MRU,\ t_y.Pos >$. Từ dòng 5 đến 9, xét trường hợp $Q \neq \emptyset$ nghĩa là C-List đang kết hợp ở mức k>2, khi đó tìm bộ $t_q \in Q$ sao cho $t_q.Tid = t_x.Tid$ và giảm $t_{x,y}.U$ một lượng bằng $t_q.U$ sau đó thêm $t_{x,y}$ vào C-List kết quả Z. Từ dòng 10 đến 13, thực hiện chiến lượng tỉa LA [7] bằng cách kiểm tra nếu không tồn tại $t_y \in Y$ có cùng Tid với t_x thì giảm giá trị MAU_{LA} một lượng bằng $MAU(t_x)$. Trường hợp $MAU_{LA} < minUtil$ thì trả về giá trị $\emptyset$ nghĩa là ngừng kết hợp và mọi tập mở rộng từ tập kết hợp sẽ không phải là tập $CoHAU$. Dòng 17 trả về C-List kết quả Z sau khi kết hợp từ X và Y.

Thuật toán *CoHAUConstruct*

Đầu vào: Một C-List Q làm tiền tố, hai C-List cần kết hợp X và Y, ngưỡng $minUtil$.

Ra: C-List Z là kết quả sau khi kết hợp X và Y

1. $MAU_{LA} = MAU(X)$,
2. **foreach** t_x *in* X **then**
3. **if** ($\exists\ t_y$ in Y sao cho $t_y.Tid = t_x.Tid$) **then**
4. $t_{x,y} \leftarrow < t_x.Tid,\ t_x.U + t_y.U,\ t_y.MRU,\ t_y.Pos >$
5. **if** $Q \neq \emptyset$ **then**
6. Tìm bộ $t_q \in Q$ với $t_q.Tid = t_x.Tid$,
7. $t_{xy}.U = t_{xy}.U - t_p.U$,
8. **end if**
9. $Z \leftarrow t_{x,y}$,
10. **else**
11. $MAU_{LA} = MAU_{LA} - MAU(t_x)$,
12. **if** $MAU_{LA} < minUtil$ **then**
13. return $\emptyset$,
14. **end if**
15. **end if**
16. **end for**
17. return Z,

5. THỰC NGHIỆM

5.1. Dữ liệu thực nghiệm

Chúng tôi cài đặt thực nghiệm thuật toán $CHAU$ bằng ngôn ngữ Java, hệ điều hành Windows 11. Thiết bị phần cứng là máy tính Dell, model Inspiron 5430, có bộ nhớ RAM 16GB, thông số CPU là Intel Core i7-1360P 2.20 GHz. Dữ liệu thực nghiệm là các bộ dữ liệu giao dịch chuẩn được tải từ thư viện SPMF [18]. Bảng 5 mô tả chi tiết các cơ sở dữ liệu thực nghiệm gồm các thông tin như số lượng giao dịch, số các mục phân biệt (I), độ dài trung bình (A) và độ dày (A/I) %. Hiệu suất thực thi của thuật toán $CHAU$ được đánh giá so với thuật toán cơ sở là $CoHAI$ [11]. Hai độ đo dùng để so sánh là thời gian thực thi (giây) và mức độ sử dụng bộ nhớ (MB).

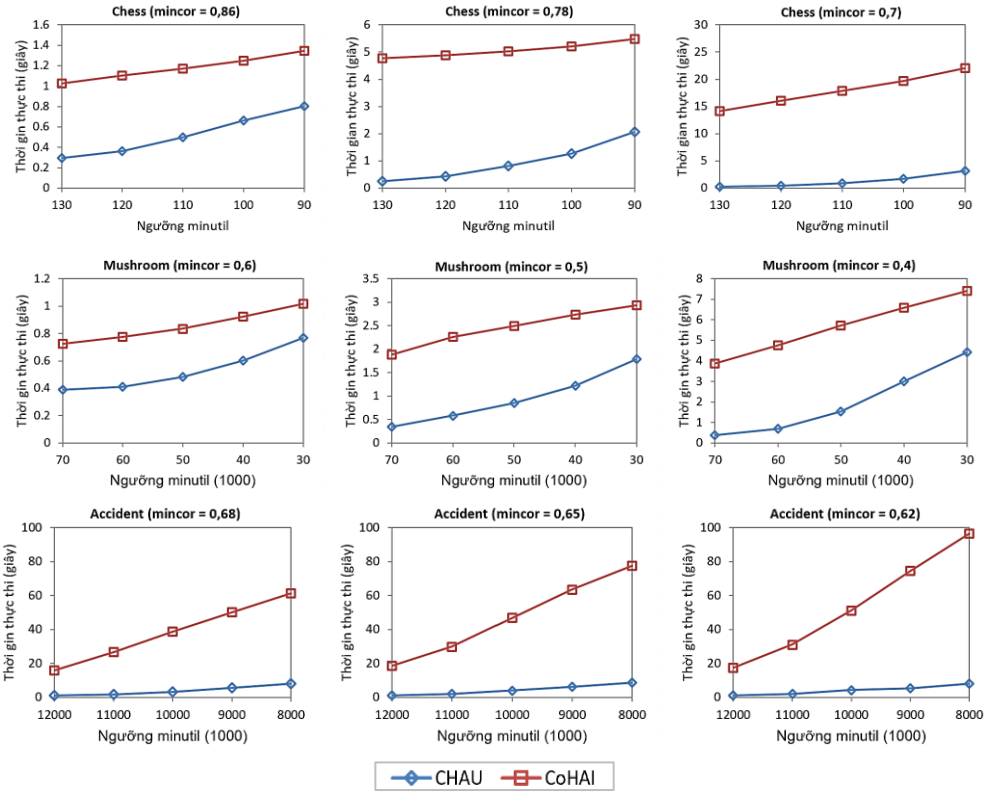
Bảng 5. Cơ sở dữ liệu thực nghiệm

Cơ sở dữ liệu	Số lượng giao dịch	Số lượng item (I)	Độ dài trung bình (A)	Độ dày (A/I) %
Chess	3.196	75	37	49,3333
Mushroom	8.124	119	23	19,3277
Kosarak	990.002	41.270	8,1	0,0196
Accident	340.183	468	33,8	7,2222
Retail	88.162	16.470	10,3	0,0625
Chainstore	1.112.949	46.086	7,23	0,0158

5.2. Kết quả thực nghiệm

Hình 3, 4 trình bày kết quả so sánh thời gian thực thi (giây) của phương pháp đề xuất khi thực thi thuật toán $CHAU$ so với thuật toán cơ sở $CoHAI$. Cả hai thuật toán đều thực nghiệm trên cùng các cơ sở dữ liệu

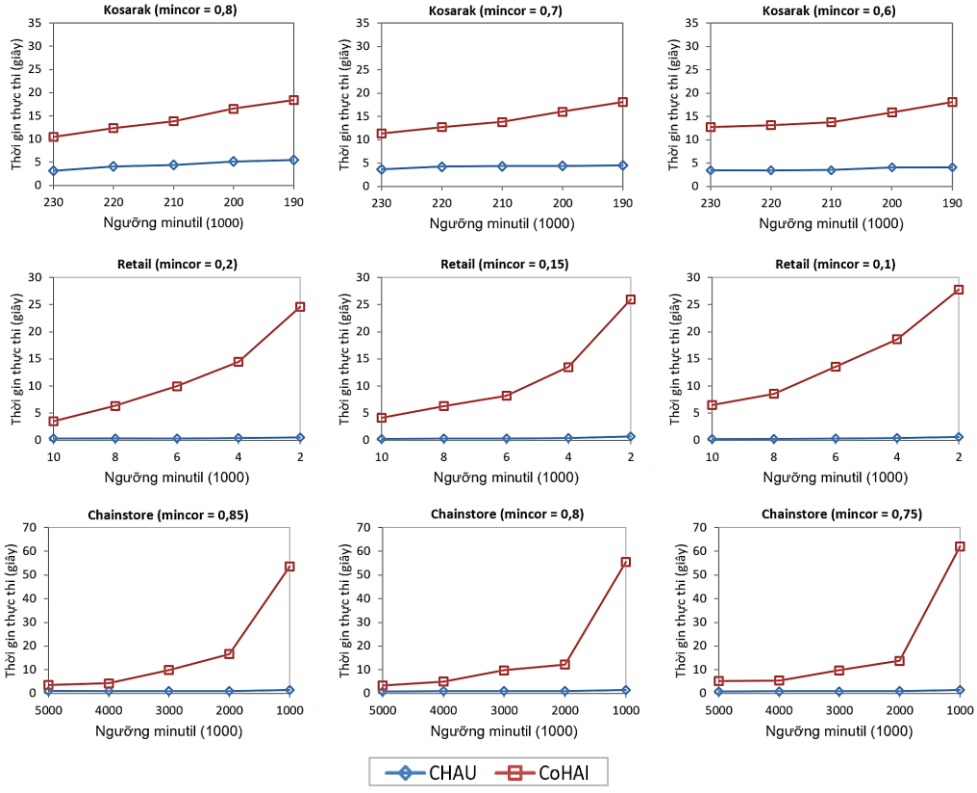
và cùng giá trị các ngưỡng. Các bộ dữ liệu được chia thành 2 nhóm dựa vào độ dày như trong Bảng 5. Trong đó, Accident, Mushroom và Chess thuộc nhóm dữ liệu dày, trong khi Chainstore, Kosarak và Retail thuộc nhóm dữ liệu thưa. Với cơ sở dữ liệu Chess (rất dày) thì thuật toán $CHAU$ thực hiện nhanh hơn $CoHAI$ trên cả 3 ngưỡng minCor là 0,86, 0,78 và 0,7. Cụ thể, khi minCor ở mức 0,86, thì thời gian thực thi của $CHAU$ tốt hơn $CoHAI$ trung bình 2,5 lần. Ở mức minCor=0,78 là 8,7 lần và minCor=0,7 là 25,8 lần. Như vậy với cùng các ngưỡng minUtil, giá trị minCor càng giảm thì thuật toán $CHAU$ thực thi càng vượt trội hơn $CoHAI$. Với Mushroom (có độ dày vừa), thời gian thực thi của thuật toán $CHAU$ cũng vượt trội hơn $CoHAI$ trên mọi giá trị ngưỡng thiết lập của minUtil và minCor. Ở ngưỡng minCor=0,4, thì thuật toán $CHAU$ chỉ mất 0,4 giây với ngưỡng minUtil=70.000 và 4,5 giây cho ngưỡng minUtil = 30.000. Trong khi thuật toán CoHAI cần đến 3,8 giây cho ngưỡng minUtil=70.000 và 7,5 giây cho ngưỡng minUtil=30.000. Với Accident, tuy có độ dạy thấp hơn nhưng số giao dịch lớn hơn rất nhiều so với Mushroom và Chess thì $CHAU$ cho kết quả thực thi vượt trội hơn thuật toán $CoHAI$. Tại ngưỡng minCor=0,68, minUtil=12.000.000 thì $CHAU$ thực thi chỉ trong 1,1 giây, trong khi thuật toán CoHAI tốn 16,2 giây. Khi giảm ngưỡng minCor còn 0,62 và minUtil=8.000.000 thì $CHAU$ thực hiện trong 8,2 giây, trong khi thuật toán $CoHAI$ thực hiện 96,5 giây. Kết quả trên đã chứng tỏ việc cải tiến phương pháp tính giá trị MAU kết hợp nhiều chiến lược tỉa đã làm tăng đáng kể hiệu suất thực thi của thuật toán.



Hình 3. Đánh giá thời gian thực thi trên nhóm dữ liệu dày

Đánh giá thời gian thực hiện trên nhóm các cơ sở dữ liệu thưa như Chainstore, Retail và Kosarak (Hình 4) cho thấy thuật toán đề xuất có hiệu suất vượt trội hơn so với thuật toán cơ sở. Với Kosarak là cơ sở dữ liệu có số mục phân biệt và số giao dịch tương đối lớn, thì thời gian chạy của thuật toán $CHAU$ đều ít hơn $CoHAI$ trên tất cả các giá trị minCor và minUtil. Tại ngưỡng minCor=0,8, minUtil=230.000 thì thuật toán $CHAU$ nhanh hơn 3,3 lần so với $CoHAI$. Khi giảm ngưỡng minCor=0,6, minUtil=190.000 thì thuật toán $CHAU$ thực thi nhanh hơn thuật toán $CoHAI$ 4,5 lần. Với Retail, một cơ sở dữ liệu thưa có số giao dịch cũng như số mục phân biệt ở mức trung bình thì thuật toán $CHAU$ cũng chứng tỏ hiệu suất tốt hơn $CoHAI$ trên mọi ngưỡng thực nghiệm. Tại ngưỡng minUtil=10.000 và minCor=0,2, thì thuật toán CHAU thực thi 0,32 giây trong khi thuật toán CoHAI thực hiện trong 3,5 giây. Tại ngưỡng minCor=0,1, minUtil=2.000, thuật toán $CHAU$ thực hiện trong 0,67 giây trong khi thuật toán $CoHAI$ cần 27,74 giây. Với tập dữ liệu lớn, rất thưa cũng như số mục phân biệt và số giao dịch cao nhất trong số các cơ sở dữ liệu thực nghiệm như Chainstore thì thuật toán $CHAU$ cho kết quả thực thi vượt trội hơn thuật toán $CoHAI$. Khi ngưỡng minUtil và minCor càng giảm thì mức chênh lệch

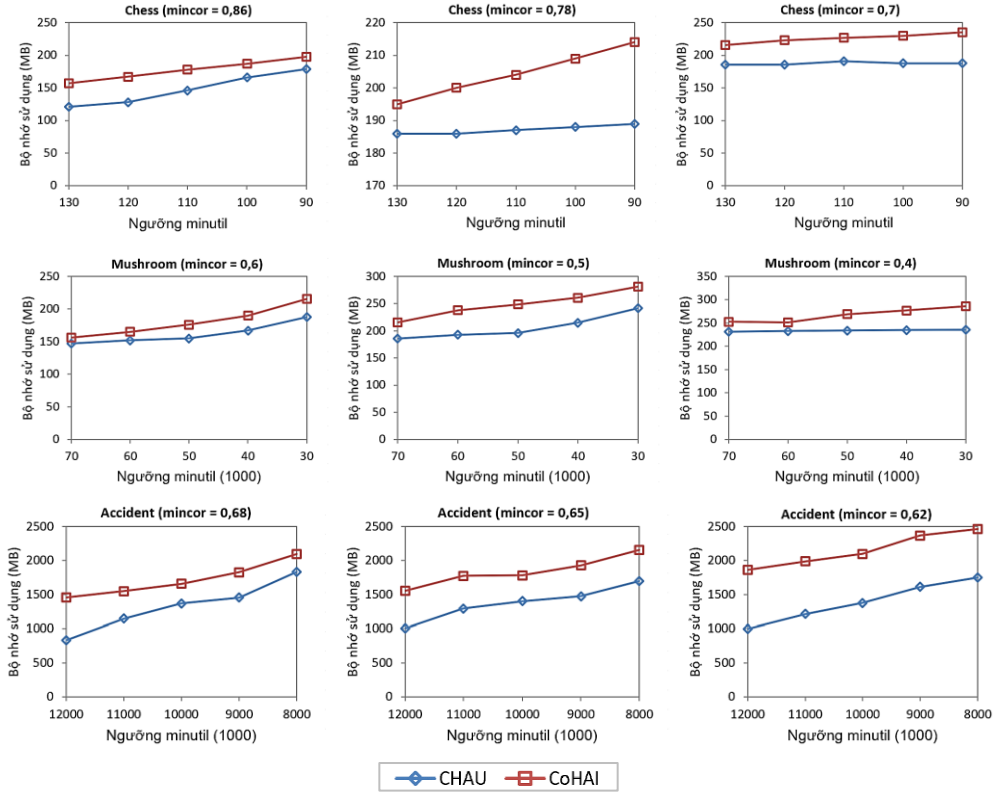
về thời gian chạy giữa hai thuật toán càng lớn. Cụ thể tại ngưỡng minCor=0,85, minUtil=5.000.000, thuật toán *CHAU* thực hiện trong 0,95 giây so với 3,47 giây của thuật toán *CoHAI*. Với ngưỡng minCor=0,75, minUtil=1.000.000, *CHAU* chỉ mất 1,42 giây so với mức 62,3 giây của *CoHAI*. Kết quả trên đã khẳng định phương pháp chúng tôi đề xuất có hiệu suất tốt hơn đáng kể so với phương pháp dùng làm cơ sở để so sánh, đặt biệt vượt trội trên các tập dữ liệu thưa và ngưỡng càng nhỏ thì thuật toán CHAU thực hiện càng hiệu quả.



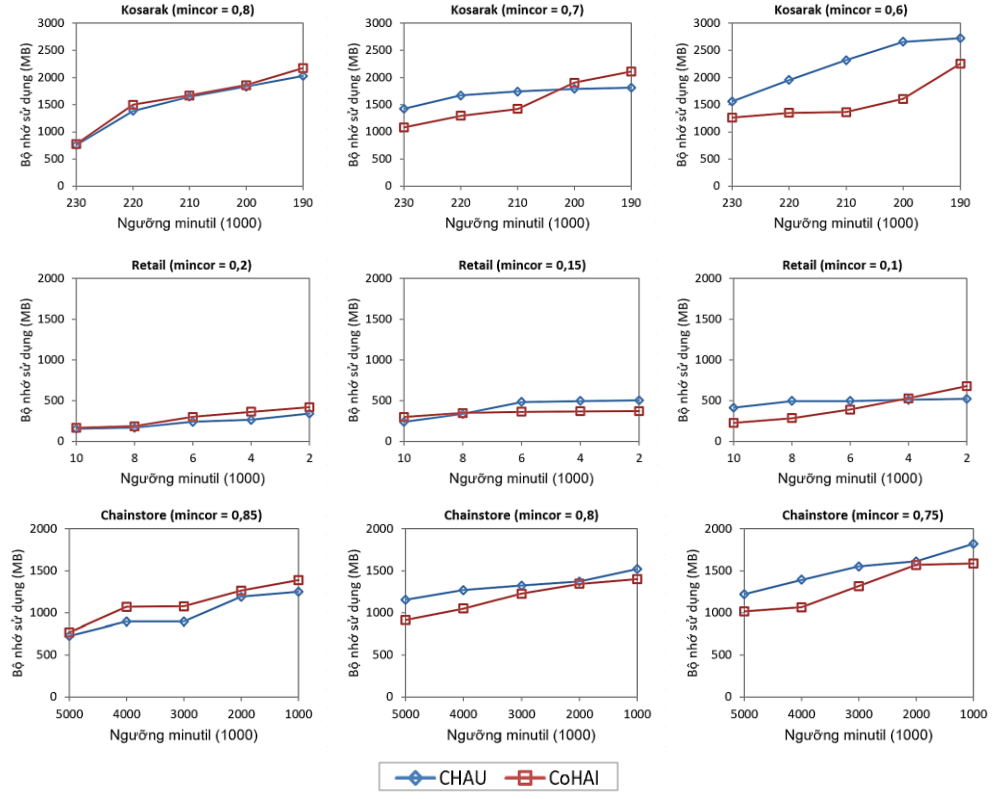
Hình 4. Đánh giá thời gian thực thi trên nhóm dữ liệu thưa

Hình 5 đánh giá mức độ sử dụng bộ nhớ của hai thuật toán so sánh trên các tập dữ liệu dày như Accident, Mushroom và Chess. Kết quả cho thấy thuật toán *CHAU* sử dụng bộ nhớ ít hơn thuật toán *CoHAI* trên cả 3 tập dữ liệu này. Đối với với tập dữ liệu Chess, mức độ sử dụng bộ nhớ của thuật toán CHAU tại ngưỡng minUtil=130.000 và minCor=0,86 là 121 MB, thuật toán *CoHAI* sử dụng 157 MB. Tại ngưỡng thấp nhất với minCor=0,7 và minUtil=90.000 thì thuật toán chúng tôi đề xuất sử dụng 188 MB dung lượng bộ nhớ so với 235 MB dung lượng bộ nhớ của thuật toán *CoHAI*. Tương tự với Mushroom, thuật toán *CHAU* cũng tiêu tốn bộ nhớ ít hơn từ 9 MB ở minCor=0,6, minUtil=70.000 đến 50 MB tại ngưỡng minCor=0,4 và minUtil=30.000. Với tập dữ liệu Accident, mức độ sử dụng bộ nhớ của *CHAU* cũng tốt hơn *CoHAI* trung bình từ 390 MB tại ngưỡng minCor=0,68 đến 765 MB tại ngưỡng minCor=0,62.

Với các tập dữ liệu thưa như Chainstore, Retail và Kosarak, mức độ sử dụng bộ nhớ được thể hiện trong Hình 6. Các tập dữ liệu này có một số đặc điểm tương đồng như số giao dịch lớn, chẳng hạn như Retail có 88.162 giao dịch, Kosarak có đến 990.002 giao dịch, Chainstore lớn nhất với 1.112.949 giao dịch. Ngoài ra số mục khác nhau trong cơ sở dữ liệu cũng lớn hơn nhiều so với các cơ sở dữ liệu dày như Kosarak có 41.270 mục, Retail có 16.470 mục, Chainstore có 46.086 mục trong khi các cơ sở dữ liệu dày có số mục cao nhất chỉ 468. Do đó khi sử dụng chiến lược tỉa EUCS trong thuật toán *CHAU* bằng cấu trúc mảng 2 chiều dẫn đến tốn bộ nhớ hơn *CoHAI* ở ngưỡng minUtil và minCor thấp. Khi giá trị ngưỡng tăng lên thì mức độ sử dụng bộ nhớ của cả hai thuật toán chênh lệch không đáng kể. Cụ thể trên tập dữ liệu Kosarak, khi giá trị minCor là 0,8 thì bộ nhớ sử dụng của cả hai thuật toán gần như tương đương ở tất cả các ngưỡng minUtil. Khi giảm ngưỡng minCor xuống 0,6 thì mức độ sử dụng bộ nhớ của *CoHAI* thấp hơn hơn *CHAU* từ 298 MB đến 466 MB. Thực nghiệm trên Retail cho thấy kết quả chiếm dụng bộ nhớ giữa hai thuật toán không chênh lệch nhiều tại hầu hết các ngưỡng minCor và minUtil. Đối với tập dữ liệu Chainstore, do có số mục lớn nên thuật toán *CHAU* sẽ tốn bộ nhớ hơn *CoHAI* ở các ngưỡng minCor thấp khi tổ chức cấu trúc EUCS. Mức chênh lệnh bộ nhớ trung bình từ 140 MB ở ngưỡng minCor =0,8 đến 209 MB ở ngưỡng minCor=0,75.



Hình 5. Đánh giá mức độ sử dụng bộ nhớ trên nhóm dữ liệu dày



Hình 6. Đánh giá mức độ sử dụng bộ nhớ trên nhóm dữ liệu thưa

5.3. Thảo luận

Trong thuật toán chúng tôi đề xuất, một số điểm có thể được cải tiến tốt hơn như sau: độ hữu ích lớn nhất phía sau tập mục $X \subset T_j$ (Maximum Remain Utility - MRU) có thể cải tiến bằng cách lấy trung bình độ hữu ích của hai mục sau X trong T_j sao cho độ hữu ích của hai mục này là cao nhất theo như đề xuất của Liu và cộng sự [19] thay vì chỉ lấy độ hữu ích lớn nhất, khi đó giá trị mới sẽ nhỏ hơn MRU hiện tại và tương tự cho giá trị MAU cũng nhỏ hơn giá trị MAU hiện tại. Từ đó sẽ làm tăng hiệu quả tỉa ứng viên (Chiến lược tỉa 3). Mặc khác, trong chiến lược tỉa dựa trên cấu trúc EUCS, chúng tôi đã cài đặt theo cấu trúc tự nhiên là dùng mảng hai chiều dẫn đến tốn kém bộ nhớ không cần thiết để lưu trữ các giá trị 0 trong phần nữa tam giác dưới của ma trận EUCS. Nếu tập dữ liệu có số mục phân biệt lớn thì càng lãng phí bộ nhớ sử dụng. Để cải thiện vấn đề này, sử dụng cấu trúc H-Map để lưu trữ sẽ là một lựa chọn tốt hơn.

6. KẾT LUẬN

Chúng tôi đã đề xuất thuật toán CHAU để khai thác tập CoHAU bằng cách cải tiến cách tính giá trị MAU cho tập mục, một trong những giá trị độ hữu ích chặn trên trong chiến lược tỉa không gian tìm kiếm. Cấu trúc *C-List* được sử dụng để tổ chức dữ liệu trong quá trình khai thác. Bên cạnh đó, thuật toán kết hợp nhiều chiến lược tỉa để nâng cao hiệu suất khai thác. Đánh giá kết quả trên hai độ đo là thời gian thực thi và bộ nhớ sử dụng đã chứng tỏ thuật toán $CHAU$ có hiệu suất tốt hơn thuật toán so sánh là $CoHAI$. Trên tất cả các tập dữ liệu thực nghiệm, thời gian chạy của CHAU đều vượt trội so với CoHAI. Tuy nhiên mức độ sử dụng bộ nhớ của thuật toán đề xuất chỉ hiệu quả hơn thuật toán $CoHAI$ trên nhóm các tập dữ liệu dày. Đối với các tập dữ liệu thưa có số mục phân biệt cao thì $CHAU$ tiêu tốn bộ nhớ hơn $CoHAI$ tại các giá trị ngưỡng minCor thấp.

Hướng phát triển tiếp theo là nghiên cứu giải pháp để giảm mức độ sử dụng bộ nhớ trên các tập dữ liệu thưa có số mục phân biệt lớn, đồng thời kết hợp với bài toán Top-k để trích xuất K tập CoHAU.

TÀI LIỆU THAM KHẢO

[1] Y. Liu, W.-k. Liao, and A. Choudhary, "A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets," in *Advances in Knowledge Discovery and Data Mining*, 2005: Springer Berlin Heidelberg, pp. 689-695, doi: https://doi.org/10.1007/11430919_79.

[2] V. S. Tseng, C.-W. Wu, B.-E. Shie, and P. S. Yu, "UP-Growth: an efficient algorithm for high utility itemset mining," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 253-262, doi: https://doi.org/10.1145/1835804.1835839.

[3] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 55-64, doi: https://doi.org/10.1145/2396761.2396773.

[4] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster High-Utility Itemset Mining Using Estimated Utility Co-occurrence Pruning," in *Foundations of Intelligent Systems*, Cham, 2014: Springer International Publishing, pp. 83-92, doi: https://doi.org/10.1007/978-3-319-08326-1_9.

[5] J. C.-W. Lin, P. Fournier-Viger, and W. Gan, "FHN: An efficient algorithm for mining high-utility itemsets with negative unit profits," *Knowledge-Based Systems,* vol. 111, pp. 283-298, 2016, doi: https://doi.org/10.1016/j.knosys.2016.08.022.

[6] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "EFIM: a fast and memory efficient algorithm for high-utility itemset mining," *Knowledge and Information Systems,* vol. 51, no. 2, pp. 595-625, 2017, doi: https://doi.org/10.1007/s10115-016-0986-0.

[7] S. Krishnamoorthy, "HMiner: Efficiently mining high utility itemsets," *Expert Systems with Applications,* vol. 90, pp. 168-183, 2017, doi: https://doi.org/10.1016/j.eswa.2017.08.028.

[8] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "Efficiently Mining High Average Utility Itemsets with a Tree Structure," in *Intelligent Information and Database Systems*, Berlin, Heidelberg, 2010: Springer Berlin Heidelberg, pp. 131-139, doi: https://doi.org/10.1007/978-3-642-12145-6_14.

[9] U. Yun and D. Kim, "Mining of high average-utility itemsets using novel list structure and pruning strategy," *Future Generation Computer Systems,* vol. 68, pp. 346-360, 2017, doi: https://doi.org/10.1016/j.future.2016.10.027.

[10] T. K. Pham and V. L. Nguyen, "An effective algorithm for mining high average utility itemsets," *Journal of Science and Technology on Information and Communications,* vol. 3, pp. 72-82, 2021.

[11] K. K. Sethi and D. Ramesh, "Correlated High Average-Utility Itemset Mining," in *Evolution in Computational Intelligence*, Singapore, 2021: Springer Singapore, pp. 485-497, doi: https://doi.org/10.1007/978-981-15-5788-0_47.

[12] P. Fournier-Viger, J. C.-W. Lin, Q.-H. Duong, and T.-L. Dam, "FHM+: Faster High-Utility Itemset Mining Using Length Upper-Bound Reduction," in *Trends in Applied Knowledge-Based Systems and Data Science*, Cham, 2016: Springer International Publishing, pp. 115-127, doi: https://doi.org/10.1007/978-3-319-42007-3_11.

[13] T. P. Hong, C. H. Lee, and S. L. Wang, "Mining high average-utility itemsets," in *2009 IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 2526-2530, doi: https://doi.org/10.1109/ICSMC.2009.5346333.

[14] J. C.-W. Lin, T. Li, P. Fournier-Viger, T.-P. Hong, J. Zhan, and M. Voznak, "An efficient algorithm to mine high average-utility itemsets," *Advanced Engineering Informatics,* vol. 30, no. 2, pp. 233-243, 2016, doi: https://doi.org/10.1016/j.aei.2016.04.002.

[15] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and H. Fujita, "Extracting non-redundant correlated purchase behaviors by utility measure," *Knowledge-Based Systems,* vol. 143, pp. 30-41, 2018, doi: https://doi.org/10.1016/j.knosys.2017.12.003.

[16] W. Gan, J. C.-W. Lin, H.-C. Chao, H. Fujita, and P. S. Yu, "Correlated utility-based pattern mining," *Information Sciences,* vol. 504, pp. 470-486, 2019, doi: https://doi.org/10.1016/j.ins.2019.07.005.

[17] B. Vo *et al.*, "Mining Correlated High Utility Itemsets in One Phase," *IEEE Access,* vol. 8, pp. 90465-90477, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2994059.

[18] P. Fournier-Viger *et al.*, "The SPMF Open-Source Data Mining Library Version 2," in *Machine Learning and Knowledge Discovery in Databases*, Cham, 2016: Springer International Publishing, pp. 36-40, doi: https://doi.org/10.1007/978-3-319-46131-1_8.

[19] X. Liu, G. Chen, F. Wu, S. Wen, and W. Zuo, "Mining top-k high average-utility itemsets based on breadth-first search," *Applied Intelligence,* vol. 53, no. 23, pp. 29319-29337, 2023, doi: https://doi.org/10.1007/s10489-023-05076-4.

ABSTRACT

MINING HIGH AVERAGE UTILITY ITEMSETS WITH CORRELATION CONSTRAINT

Nguyen Thi Thanh Thuy, Nguyen Van Le*, Manh Thien Ly

Ho Chi Minh City University of Industry and Trade

*Email: *lenv@huit.edu.vn*

High utility itemset is a crucial concept in the itemset mining problem. To address the imbalance in utility between elements within an itemset, the concept of average utility has been proposed. However, some itemsets with high average utility still exhibit low element correlation, reducing their value for business analysis. To overcome this limitation, this paper proposes a method for mining high average utility itemsets that considers correlation through the CHAU (Correlated High Average Utility) algorithm. The research focuses on improving the formula for calculating the upper bound of average utility to increase candidate pruning capability, thereby improving the algorithm's processing performance. Experimental results comparing the proposed method with the state-of-the-art CoHAI algorithm across datasets with varying sparsity and density, including Chainstore, Kosarak, Retail, Accident, Mushroom, and Chess, show that the proposed method achieves better performance in both execution time and memory consumption.

Keywords: Correlation measure, correlated high-average utility itemset, CoHAU exploitation, correlation constraint in itemset, minimum utility threshold.